UECM1703 Introduction of Scientific Computing Marking Guide

TOPIC 6 Practical

UNIVERSITI TUNKU ABDUL RAHMAN

Faculty:	FES	Unit Code:	UECM1703
Course:	AM &FM	Unit Title:	Introduction To Scientific Computing
Year:	1&2	Lecturer:	Dr Yong Chin Khian
Session:	Oct 2022		

Q1. A local Bank wants to estimate the number of foreclosures per 1000 houses sold. Bank officials think that the number of foreclosures is related to the size of the down payment made by house buyers. The following table contains foreclosure and down payment data.

Down Payment Size (% of purchase price)	Number of Foreclosures per 1000 houses
10	42
20	11
14	29
12	25
18	12
16	24

You fit the above data to $y = \beta_0 + \beta_1 x + \epsilon$, where y is the number of foreclosures per 1000 houses sold, and x is the size of the down payment made by house buyers. Use the stat.models module to compute the adjusted R^2.

Ans.

Adjusted $R^2 = 0.8168756061640055$

```
import numpy as np
import statsmodels.api as sm
y = np.array([42,11,29,25,12,24])
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([10,20,14,12,18,16]).reshape(-1,1)
x = np.hstack((x0, x1))
model = sm.OLS(y, x)
results = model.fit()
print("Adj. R-squared = ", results.rsquared_adj)

Output:
Adj. R-squared =  0.8168756061640144
```

<u>**UECM1703 Introduction of Scientific Computing Marking Guide**</u>

Q2. Consider the data shown below:

y	x	y	x
1791	16	971	13
450	10	594	11
594	11	1791	16
331	9	103	6
1208	14	594	11
103	6	594	11
2143	17	1480	15
450	10	331	9

Fit the polynomial regression models up to order 4. A model that best fit the data based on the AIC criteria was fitted, determine the fitted value of y when $x = 10$.

Ans.

Since the 3-order polynomial has the smallest AIC, thus, model with polynomial 3-th degree should be selected. So,
$\hat{y} = -0.01 + 0.75x0.21x^2 + 0.4212x^3$
when $x = 10$,
$\hat{y} = -0.01 + 0.75(10)0.21(10)^2 + 0.4212(10)^3 = 449.73$

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
import statsmodels.api as sm
y = np.array([1791,450,594,331,1208,103,2143,450,
971,594,1791,103,594,594,1480,331])
x = np.array([16,10,11,9,14,6,17,10,
13,11,16,6,11,11,15,9]).reshape((-1,1))
x1st = PolynomialFeatures(degree=1, include_bias=True).fit_transform(x)
x2nd = PolynomialFeatures(degree=2, include_bias=True).fit_transform(x)
x3rd = PolynomialFeatures(degree=3, include_bias=True).fit_transform(x)
x4th = PolynomialFeatures(degree=4, include_bias=True).fit_transform(x)
model1 = sm.OLS(y, x1st)
results1 = model1.fit()
model2 = sm.OLS(y, x2nd)
results2 = model2.fit()
model3 = sm.OLS(y, x3rd)
results3 = model3.fit()
model4 = sm.OLS(y, x4th)
results4 = model4.fit()
print("AIC model1 = ", results1.aic)
print("AIC model2 = ", results2.aic)
print("AIC model3 = ", results3.aic)
print("AIC model4 = ", results4.aic)
print("Beta hat = ", results4.params)
xnew = np.array([1., 10, 10**2, 10**3])
ynew = results3.predict(xnew)
print(ynew)
Output:
AIC model1 =  212.23897720951726
AIC model2 =  131.1878105377471
AIC model3 =  8.969388437525602
AIC model4 =  9.331420776455019
Beta hat =  [-0.011561530764993222  0.7488346050241717
0.21051398970421076 0.42120364877870653]
y_h= [[449.73183227]]
```

Q3. Consider the data shown below:

y	x	y	x
1860	16	1006	13
465	10	615	11
615	11	1860	16
341	9	106	6
1252	14	615	11
106	6	615	11
2226	17	1536	15
465	10	341	9

Fit the polynomial regression models up to order 4. A model that best fit the data based on the BIC criteria was fitted, determine the corresponding BIC values.

Ans.

Since the 4-order polynomial has the smallest BIC, thus, model with polynomial forth degree should be selected. So,
$\hat{y} = 21.84 + -7.49x1.28x^2 + 0.3765x^3 + 0.3765x^4$
Min BIC $= 9.475833803668186$

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
import statsmodels.api as sm
y = np.array([1860,465,615,341,1252,106,2226,
465,1006,615,1860,106,615,615,1536,341])
```

```
x = np.array([16,10,11,9,14,6,17,10,
13,11,16,6,11,11,15,9]).reshape((-1,1))
x1st = PolynomialFeatures(degree=1, include_bias=True).fit_transform(x)
x2nd = PolynomialFeatures(degree=2, include_bias=True).fit_transform(x)
x3rd = PolynomialFeatures(degree=3, include_bias=True).fit_transform(x)
x4th = PolynomialFeatures(degree=4, include_bias=True).fit_transform(x)
model1 = sm.OLS(y, x1st)
results1 = model1.fit()
model2 = sm.OLS(y, x2nd)
results2 = model2.fit()
model3 = sm.OLS(y, x3rd)
results3 = model3.fit()
model4 = sm.OLS(y, x4th)
results4 = model4.fit()
print("BIC model1 = ", results1.bic)
print("BIC model2 = ", results2.bic)
print("BIC model3 = ", results3.bic)
print("BIC model4 = ", results4.bic)
Min_BIC = min(results1.bic, results2.bic, results3.bic, results4.bic)
print("Min BIC = ", Min_BIC)
Output:
BIC model1 =  215.13833036546802
BIC model2 =  134.83402920847652
BIC model3 =  9.648106744351137
BIC model4 =  9.475833803668186
Min BIC  =  9.475833803668186
```

Q4. Consider the data shown below:

y	24	25	19	29	20	23	26	21	21	23	26	31
x	12	13	8	17	9	11	14	10	10	11	14	19

You fit the above data to $Y = \beta_0 + \beta_1 X + \epsilon$. Obtain a prediction for a new observation y_h when $x_h = 18.0$.

Ans.

(a) $\hat{y} = 10.3629 + 1.1057x$

(b) $y_h = 10.3629 + 1.1057(18.0) = 30.2657$

```
Python code:
import numpy as np
import statsmodels.api as sm
y = np.array([24,25,19,29,20,23,26,21,21,23,26,31])
n = len(y)
x0 = np.ones(n).reshape((-1,1))
x1 = np.array([12,13,8,17,9,11,14,10,10,11,14,19]).reshape((-1,1))
x = np.hstack((x0,x1))
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())
coef = results.params
print(coef)
xnew = np.array([1,18.0])
ynew = results1.predict(xnew)
print("Ynew=",ynew)
Output:
Beta= [10.36 1.11]
Ynew= [30.2657]
```

Q5. You fit the following data to $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$ with independent normal error terms.

y	x_1	x_2	x_3	y	x_1	x_2	x_3
105.6	7.9	46.2	15.4	86.8	4.6	23.9	8.7
87.4	4.7	24.7	8.9	82.9	3.9	19.3	7.3
99.2	6.8	38.7	13.1	72.4	2.0	7.0	3.6
106.5	8.1	47.4	15.7	85.5	4.4	22.4	8.2
94.7	6.0	33.2	11.5	89.8	5.1	27.5	9.8
104.3	7.7	44.6	14.9	74.5	2.4	9.4	4.3
89.1	5.0	26.8	9.5	88.6	4.9	26.2	9.3
101.5	7.2	41.3	13.9	82.7	3.9	19.0	7.2

Use the stat.models package in Python to obtain a prediction for a new observation Y_h when $x_{h_1} = 5.9$, $x_{h_2} = 30.8$ and $x_{h_3} = 10.2$.

Ans.

$$\hat{Y} = 62.9413 + 1.5624(5.9) + 0.1943(30.8) + 1.3848(10.2) = 92.2675$$

```
import numpy as np
import statsmodels.api as sm
y = np.array([105.6, 86.8,87.4,82.9,99.2,72.4,106.5,85.5,
              94.7,89.8,104.3,74.5,89.1,88.6,101.5, 82.7])
n = len(y)
```

```
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([7.9, 4.6,4.7,3.9,6.8,2.0,8.1,4.4,
               6.0,5.1,7.7,2.4,5.0,4.9,7.2,3.9]).reshape(-1,1)
x2 = np.array([46.2, 23.9,24.7,19.3,38.7,7.0,47.4,22.4,
               33.2,27.5,44.6,9.4,26.8,26.2,41.3,19.0]).reshape(-1,1)
x3 = np.array([15.4, 8.7,8.9,7.3,13.1,3.6,15.7,8.2,
               11.5,9.8,14.9,4.3,9.5,9.3,13.9,7.2]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3))
model = sm.OLS(y, x)
results = model.fit()
#print(results.summary())
Beta = results.params
print("Beta hat = ", Beta)
x_new = np.array([1,5.9,30.8,10.2]).reshape((-1,4))
print("x_new=",x_new)
y_new = results.predict(x_new)
print("y_new=",y_new)
Output:
Beta hat =  [62.9413 1.5624 0.1943 1.3848]
x_new= [[ 1.   5.9 30.8 10.2]]
y_new= [92.2675]
```

Q6. You fit $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$ with independent normal error terms to data in data file dfQ09c.csv. Use the stat.models package in Python to answer the following questions.

(a) State the estimated regression function.

Ans.
$\hat{Y} = 64.0657 + 1.182X_1 + 0.3074X_2 + 1.2265X_3 = 2.2815X_4$

(b) Obtain a prediction for a new observation Y_h when $x_{h_1} = 5.9$, $x_{h_2} = 30.8$, $x_{h_3} = 10.2$ and $x_{h_4} = 19.0$.

Ans.
$\hat{Y} = 64.0657 + 1.182(5.9) + 0.3074(30.8) + 1.2265(10.2) + 2.2815(19.0) = 136.3666$

(c) Write down the python codes and the correspoding outputs that you obtained for answers in part (a) and part (b).

Ans.

```
Python code:
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
df = pd.read_csv('dfQ09c.csv')
y = df.y
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array(df.x1).reshape(-1,1)
x2 = np.array(df.x2).reshape(-1,1)
x3 = np.array(df.x3).reshape(-1,1)
x4 = np.array(df.x4).reshape(-1,1)
x = np.hstack((x0,x1,x2,x3,x4))
x, y = np.array(x), np.array(y)
model = sm.OLS(y, x)
results = model.fit()
Beta = results.params
print("beta = ",Beta)
x_new = np.array([1,5.9,30.8,10.2,19.0]).reshape(-1,5)
print("x_new=",x_new)
y_new = results.predict(x_new)
print("y_new=",y_new)
#print(results.summary())
Output:
Beta hat =  [64.0657 1.182 0.3074 1.2265 1.2265]
x_new= [[ 1.   5.9 30.8 10.2 19.0]]
y_new= [136.3666]
```

Q7. The quality of orange juice produced by a manufacturer is constantly monitored. Is there a relationship between the sweetness index and chemical measure such as the amount of water soluble pectin (parts per million) in the orange juice? Data collected on these two variables for 24 production runs at juice manufacturing plant are save in orjuice.csv file. Suppose a manufacturer wants to use simple linear regression to predict the sweetness(Y) from the amount of pectin (X).

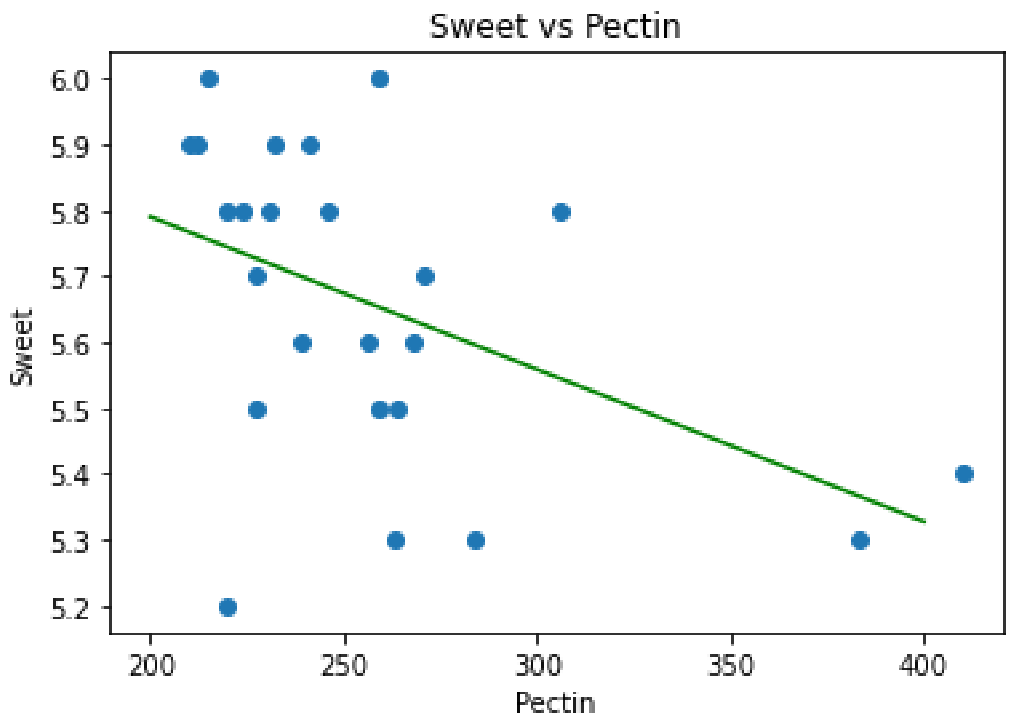
(a)Fit the data to the model $Y = \beta_0 + \beta_1 X + \epsilon$.

(b)Plot the fitted regression function and the data.

Ans.

(a)$\hat{y} = 6.252067909048172 - 0.0023106258824640955x$

(b)



```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
df = pd.read_csv('orjuice.csv')
#print(df)
model1 = np.poly1d(np.polyfit(df.Pectin, df.Sweet, 1))
print(model1)
plt.scatter(df.Pectin, df.Sweet)
plt.title("Sweet vs Pectin")
plt.xlabel("Pectin")
plt.ylabel("Sweet")
polyline = np.linspace(200, 400, 500)
plt.plot(polyline, model1(polyline), color='green')
```

Q8. Consider the data shown below:

y	x	y	x
1008246	32	377998	25
155960	20	227586	22
155960	20	888652	31
81938	17	20711	12
592936	28	189241	21
10111	10	189241	21
1139524	33	681678	29
127274	19	81938	17

Fit the polynomial regression models up to order 4, then answer the following questions:

(a)Fill in the table below:

Polynomial model	Adjusted R^2	AIC	BIC
First order			
Second order			
Third order			
Fourth order			

(b)Determine a model that best fit the data based on the adjusted R^2.

(c)Write down the equation of the best fitted curve that you have selected from part (b).

(d)Created a scatter plot of the data and add the fitted polynomial line to the scatterplot.

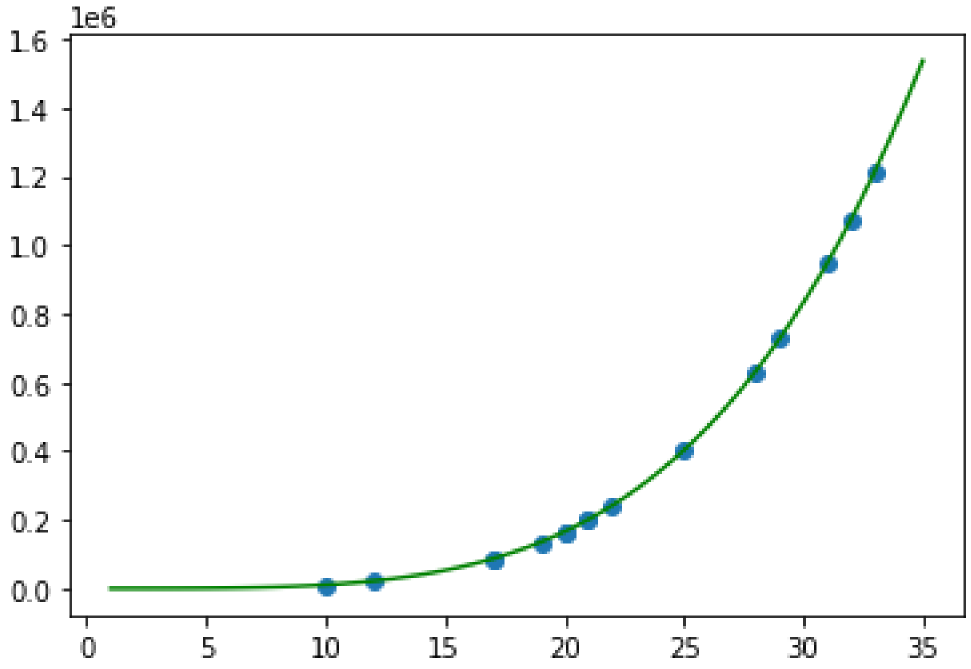
Ans.

(a)

Polynomial model	Adjusted R^2	AIC	BIC
First order	0.861	426.1	427.6
Second order	0.996	369.1	371.4
Third order	1.0	284.2	287.3
Fourth order	1.0	12.2	16.063

(b)Since the fourth-order polynomial has the largest adjusted R^2 thus, model with polynomial fourth degree should be selected.

(c)$y = -14.9785 + 3.42x + 0.0500x^2 + 0.6875x^3 + 0.9399x^4$



(d)

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import statsmodels.api as sm
y = np.array([1,008,246,155,960,155,960,81,938,592,936,10,111,1,139,524,127,274,
              377,998,227,586,888,652,20,711,189,241,189,241,681,678,81,938])
print(y)
x = np.array([32,20,20,17,28,10,33,19,
              25,22,31,12,21,21,29,17]]).reshape((-1,1))
x1st = PolynomialFeatures(degree=1, include_bias=True).fit_transform(x)
x2nd = PolynomialFeatures(degree=2, include_bias=True).fit_transform(x)
x3rd = PolynomialFeatures(degree=3, include_bias=True).fit_transform(x)
x4th = PolynomialFeatures(degree=4, include_bias=True).fit_transform(x)
model1 = sm.OLS(y, x1st)
results1 = model1.fit()
model2 = sm.OLS(y, x2nd)
results2 = model2.fit()
model3 = sm.OLS(y, x3rd)
results3 = model3.fit()
model4 = sm.OLS(y, x4th)
results4 = model4.fit()
print("Adj. R-squared model1 = ", results1.rsquared_adj)
print("AIC model1 = ", results1.aic)
print("BIC model1 = ", results1.bic)
print("Adj. R-squared model2 = ", results2.rsquared_adj)
print("AIC model2 = ", results2.aic)
print("BIC model2 = ", results2.bic)
print("Adj. R-squared model3 = ", results3.rsquared_adj)
print("AIC model3 = ", results3.aic)
print("BIC model3 = ", results3.bic)
print("Adj. R-squared model4 = ", results4.rsquared_adj)
print("AIC model4 = ", results4.aic)
print("BIC model4 = ", results4.bic)
print("Beta hat = ", results4.params)
#print("results1",results1.summary())
#print("Results2",results2.summary())
#print("Results3",results3.summary())
#print("results4",results4.summary())
Output:
Adj. R-squared model1 =  0.861
AIC model1 =  426.1
BIC model1 =  427.6
Adj. R-squared model2 =  0.996
AIC model2 =  369.1
BIC model2 =  371.4
Adj. R-squared model3 =  1.0
AIC model3 =  284.2
BIC model3 =  287.3
Adj. R-squared model4 =  1.0
AIC model4 =  12.2
BIC model4 =  16.063
Beta hat =  [-14.9785 3.4197 0.0500 0.6875 0.9399]
```

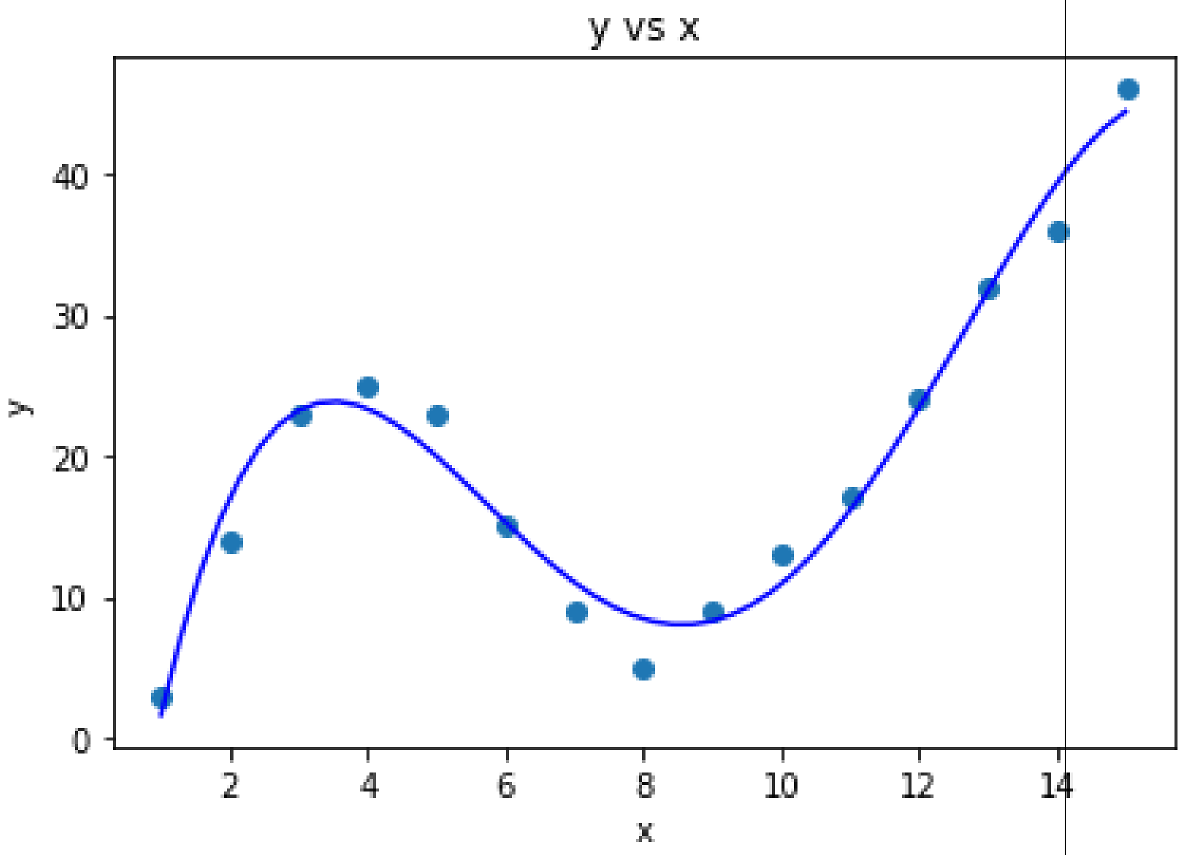
Q9. You are given the following data:

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	3	14	23	25	23	15	9	5	9	13	17	24	32	36	46

Fit the data to the polynomial regression model of degree 4. Created a scatter plot of the data and add the fitted polynomial line to scatterplot.

Ans.
```
import numpy as np
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
y = [3, 14, 23, 25, 23, 15, 9, 5, 9, 13, 17, 24, 32, 36, 46]
#fit polynomial models up to degree 5
model4 = np.poly1d(np.polyfit(x, y, 4))
#create scatterplot
polyline = np.linspace(1, 15, 50)
```

```python
plt.scatter(x, y)
plt.title("y vs x")
plt.xlabel("x")
plt.ylabel("y")
#add fitted polynomial lines to scatterplot
plt.plot(polyline, model4(polyline), color='blue')
plt.show()
```



Q10. Consider the data below:

y	x_1	x_2	x_3	x_4
268.6	7.9	31.6	15.4	30.2
181.7	4.6	29.4	8.7	16.8
184.9	4.7	29.5	8.9	17.3
164.1	3.9	28.9	7.3	14.1
239.4	6.8	30.9	13.1	25.7
116.0	2.0	27.7	3.6	6.7
273.7	8.1	31.7	15.7	31.0
175.8	4.4	29.2	8.2	15.9
218.0	6.0	30.3	11.5	22.4
195.9	5.1	29.8	9.8	19.0

Assuming that regression model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \epsilon$ is appropriate. Find the test statistic for testing $H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$.

Ans.

F = 10880869.344343184

```
import numpy as np
import statsmodels.api as sm
y = [268.6, 181.7, 184.9, 164.1, 239.4,
116.0, 273.7, 175.8, 218.0,195.9]
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([7.9, 4.6, 4.7, 3.9, 6.8,
2.0, 8.1, 4.4, 6.0, 5.1]).reshape(-1,1)
x2 = np.array([31.6, 29.4, 29.5, 28.9, 30.9,
27.7, 31.7, 29.2, 30.3, 29.8]).reshape(-1,1)
x3 = np.array([15.4, 8.7, 8.9, 7.3, 13.1,
3.6, 15.7, 8.2, 11.5, 9.8]).reshape(-1,1)
x4 = np.array([30.2, 16.8, 17.3, 14.1, 25.7,
6.7, 31.0, 15.9, 22.4, 19.0]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3,x4))
x, y = np.array(x), np.array(y)
model = sm.OLS(y, x)
results = model.fit()
Fstat = results.fvalue
print("Fstat = ",Fstat)
Output:
Fstat =  10880869.344343184
```

Q11. Consider the data below:

y	x_1	x_2	x_3	x_4
223.5	5.6	30.1	10.7	20.8
242.9	6.2	30.5	12.0	23.5
177.8	4.0	29.0	7.5	14.5
251.6	6.5	30.7	12.6	24.7
203.1	4.9	29.6	9.2	18.0
152.3	3.1	28.4	5.7	11.0
193.0	4.5	29.4	8.6	16.6
247.3	6.4	30.6	12.3	24.1
235.7	6.0	30.3	11.5	22.5
171.4	3.8	28.9	7.1	13.6

Assuming that regression model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \epsilon$ is appropriate. Find the test statistic for testing $H_0 : \beta_2 = 0$.

Ans.

t = 0.33106864409548303

```
import statsmodels.api as sm
y = [223.5, 242.9, 177.8, 251.6, 203.1,
152.3, 193.0, 247.3, 235.7,171.4]
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([5.6, 6.2, 4.0, 6.5, 4.9,
3.1, 4.5, 6.4, 6.0, 3.8]).reshape(-1,1)
x2 = np.array([30.1, 30.5, 29.0, 30.7, 29.6,
28.4, 29.4, 30.6, 30.3, 28.9]).reshape(-1,1)
x3 = np.array([10.7, 12.0, 7.5, 12.6, 9.2,
5.7, 8.6, 12.3, 11.5, 7.1]).reshape(-1,1)
x4 = np.array([20.8, 23.5, 14.5, 24.7, 18.0,
11.0, 16.6, 24.1, 22.5, 13.6]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3,x4))
x, y = np.array(x), np.array(y)
model = sm.OLS(y, x)
results = model.fit()
Tstat = results.tvalues[2]
print("Tstat = ", Tstat)
Output:
Tstat = 0.33106864409548303
```

Q12. Consider the data below:

y	x_1	x_2	x_3	x_4
206.6	5.6	30.1	10.7	20.8
223.7	6.2	30.5	12.0	23.5
166.0	4.0	29.0	7.5	14.5
231.4	6.5	30.7	12.6	24.7
188.5	4.9	29.6	9.2	18.0
143.5	3.1	28.4	5.7	11.0
179.6	4.5	29.4	8.6	16.6
227.7	6.4	30.6	12.3	24.1
217.4	6.0	30.3	11.5	22.5
160.4	3.8	28.9	7.1	13.6

Assuming that regression model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \epsilon$ is appropriate. Find the p-value for testing $H_0 : \beta_2 = 0$.

Ans.

P-Value = 0.6532412343289847

```
import statsmodels.api as sm
y = [206.6, 223.7, 166.0, 231.4, 188.5,
143.5, 179.6, 227.7, 217.4,160.4]
n = len(y)
x0 = np.ones(n).reshape(-1,1)
x1 = np.array([5.6, 6.2, 4.0, 6.5, 4.9,
3.1, 4.5, 6.4, 6.0, 3.8]).reshape(-1,1)
x2 = np.array([30.1, 30.5, 29.0, 30.7, 29.6,
28.4, 29.4, 30.6, 30.3, 28.9]).reshape(-1,1)
x3 = np.array([10.7, 12.0, 7.5, 12.6, 9.2,
5.7, 8.6, 12.3, 11.5, 7.1]).reshape(-1,1)
x4 = np.array([20.8, 23.5, 14.5, 24.7, 18.0,
11.0, 16.6, 24.1, 22.5, 13.6]).reshape(-1,1)
x = np.hstack((x0, x1,x2,x3,x4))
x, y = np.array(x), np.array(y)
model = sm.OLS(y, x)
results = model.fit()
P_value = results.pvalues[2]
print("Pvalue = ", P_value)
Output:
Pvalue = 0.6532412343289847
```

Q13. Consider the data shown below:

y	x	y	x
1065	11.2	660	10.1
449	9.3	521	9.6
449	9.3	1022	11.1
366	8.9	222	8.0
826	10.6	496	9.5
197	7.8	496	9.5
1109	11.3	901	10.8
406	9.1	347	8.8

You fit the above data to $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \beta_4 x^4 + \epsilon$.

(a) Write down the equation of the fitted curve.

Ans.

$\hat{y} = -2161.9282 + 920.86x - 143.9705x^2 + 9.3993x^3 + -0.1419x^4$

```
import numpy as np
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
y=[1065,449,449,366,826,197,1109,406,660,521,1022,222,
   496,496,901,347]
n = len(y)
x = np.array([11.2,9.3,9.3,8.9,10.6,7.8,11.3,9.1,
10.1,9.6,11.1,8.0,9.5,9.5,10.8,8.8]).reshape(-1,1)
x = PolynomialFeatures(degree=4, include_bias=True).fit_transform(x)
Mod = sm.OLS(y,x)
results = Mod.fit()
b = results.params
print("b",b)
Output:
b [-2161.9281594111235  920.8585143492382
-143.97053260946632  9.399332398341812 -0.14187999017441655]
```

(b) Is the model $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \epsilon$ significant for estimating y? Justify your answer using p-value.

Ans.

Since p-value = 6.3589808373373316e-37 < 0.05, The null hypothesis of the model not significance for estimating y is not rejected. Hence the model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \epsilon$ is not significant for estimating the texture.

```
import numpy as np
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
y=[1065,449,449,366,826,197,1109,406,660,521,1022,222,
   496,496,901,347]
n = len(y)
x = np.array([11.2,9.3,9.3,8.9,10.6,7.8,11.3,9.1,
10.1,9.6,11.1,8.0,9.5,9.5,10.8,8.8]).reshape(-1,1)
x = PolynomialFeatures(degree=4, include_bias=True).fit_transform(x)
Mod = sm.OLS(y,x)
results = Mod.fit()
pvalue = results.f_pvalue
print("Pvalue=", pvalue)
Output:
```

```
Pvalue= 6.358980837337316e-37
```

(c) Determine the predicted value of the mean of y when $x = 7.7$.

Ans.
$\hat{y} = -2161.9282 + 920.86(7.7) - 143.9705(7.7^2) + 9.3993(7.7^3) + -0.1419(7.7^)4 = 185.02362930685013$

```
Beta = Results.params
xh = np.array([1,7.7,59.290000000000006,456.533, 3515.3041000000003])
yh = np.inner(xh, Beta)
print("yh=",yh)

Output:
yh = 185.02362930685013
```

(d) Created a scatter plot of the data with the fitted polynomial line added to the scatterplot.

Ans.

```
#create scatterplot
dfx = [11.2,9.3,9.3,8.9,10.6,7.8,11.3,9.1,
10.1,9.6,11.1,8.0,9.5,9.5,10.8,8.8]
plt.scatter(x, y)
Moda = np.poly1d(np.polyfit(dfx, y, 4))
polyline = np.linspace(1, 12, 50)
#add fitted polynomial lines to scatterplot
plt.plot(polyline, Moda(polyline), color='green')
pli.show()
```